МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**РУТ (МИИТ)**

**Кафедра «Управление и защита информации»**

КУРСОВОЙ ПРОЕКТ

по дисциплине

«Системы искусственого интеллекта»

на тему:

*«Решение задачи регрессии: предсказание медицинских расходов»*

Выполнил: студент гр. УВП-412 Рогов К.Д.

Принял: доцент Зольникова Н. Н.

Москва – 2025

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#__RefHeading___Toc8629_1812597043)

[1 Предварительная обработка данных и постановка задачи 4](#__RefHeading___Toc8631_1812597043)

[1.1 Описание набора данных 4](#__RefHeading___Toc8633_1812597043)

[1.2 Подготовка данных 4](#__RefHeading___Toc8635_1812597043)

[2 Реализация регрессии двумя способами 10](#__RefHeading___Toc8637_1812597043)

[2.1 Модель без использования библиотек 10](#__RefHeading___Toc8639_1812597043)

[2.2 Модель с использованием библиотек 16](#__RefHeading___Toc8641_1812597043)

[3 Разработка программы для работы с обученной моделью 20](#__RefHeading___Toc8643_1812597043)

[Заключение 26](#__RefHeading___Toc8647_1812597043)

[Приложение А 27](#__RefHeading___Toc8649_1812597043)

# **ВВЕДЕНИЕ**

В данном работе реализована задача предсказания страховых расходов на основе набора данных «insurance». Решение задачи выполнено двумя способами:

* **Модель без использования библиотек** – реализована с нуля с помощью метода градиентного спуска.
* **Модель с использованием библиотек** – использует готовые инструменты, в частности класс LinearRegression из scikit-learn.

Кроме того, разработана программа с графическим интерфейсом (GUI), которая позволяет пользователю ввести исходные данные и получить предсказание расходов. Отчет включает анализ предварительной обработки данных, описание моделей, визуализацию результатов и разъяснение нюансов реализации.

# **1. Предварительная обработка данных и постановка задачи**

## 1.1 Описание набора данных

Набор данных «insurance.csv» содержит информацию о клиентах и их страховых расходах. Основные признаки включают:

* **age** – возраст клиента,
* **sex** – пол (значения «male» и «female»),
* **bmi** – индекс массы тела,
* **children** – количество детей,
* **smoker** – наличие привычки курения («yes» или «no»),
* **region** – регион проживания.

Целевая переменная – **charges** – представляет собой затраты на страхование. В целях стабилизации разброса значений и улучшения качества модели, значения колонки преобразуются с помощью логарифма.

## 1.2 Подготовка данных

Для корректной работы моделей были выполнены следующие этапы подготовки данных:

* **Кодирование категориальных признаков.**  
  Признаки sex, smoker и region преобразуются в числовые значения. Например, для smoker используется отображение:

df['sex'] = df[‘smoker'].map({‘yes': 1, ‘no': 0})

Аналогичным образом кодируются и другие категориальные переменные.

* **Нормализация числовых признаков.**  
  Для признаков age, bmi и children применяется стандартизация:

for col in ['age', 'bmi', 'children']:

mean\_val = df[col].mean()

std\_val = df[col].std()

df[col] = (df[col] - mean\_val) / std\_val

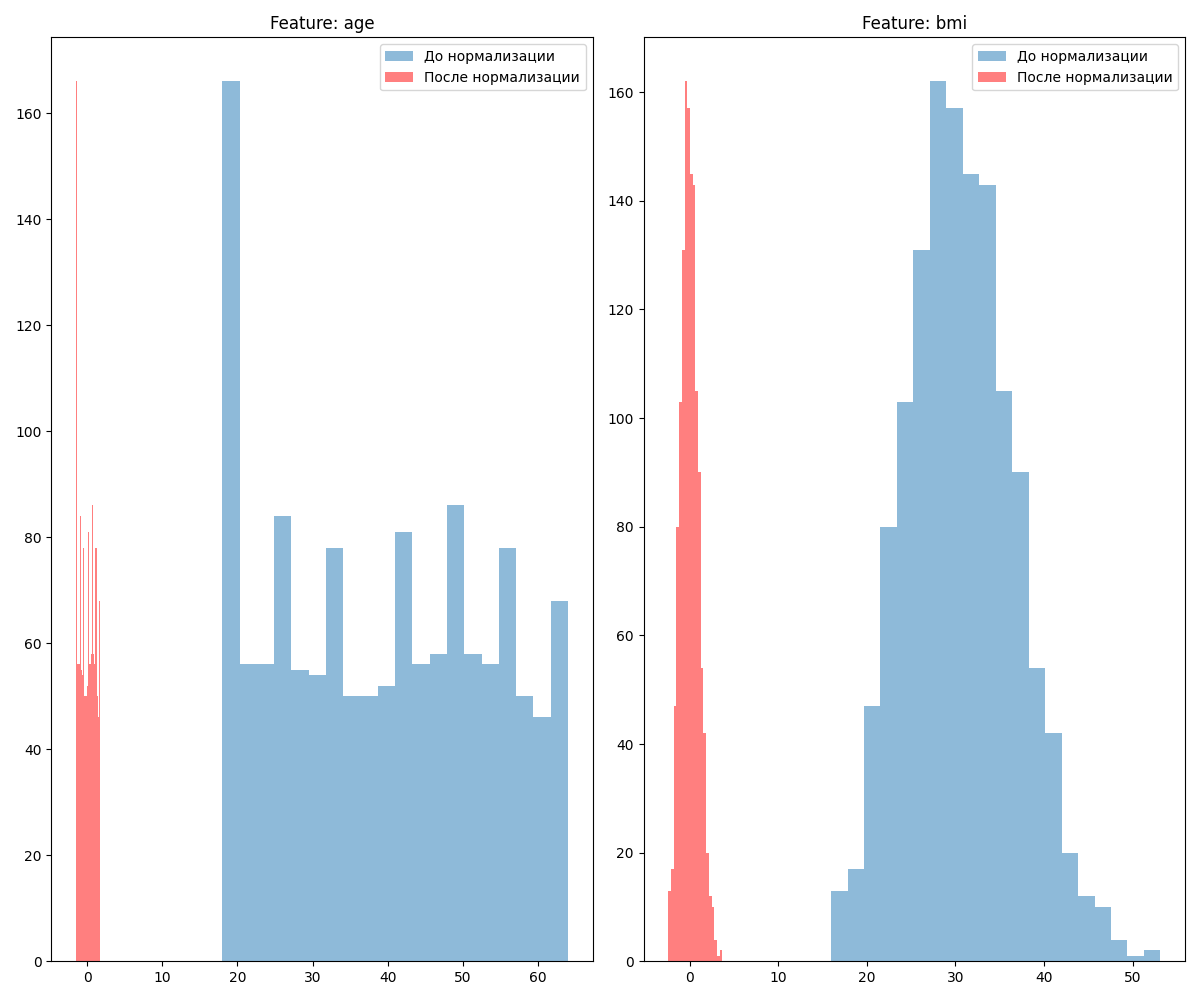
Это помогает модели быстрее сходиться и улучшает интерпретируемость коэффициентов.

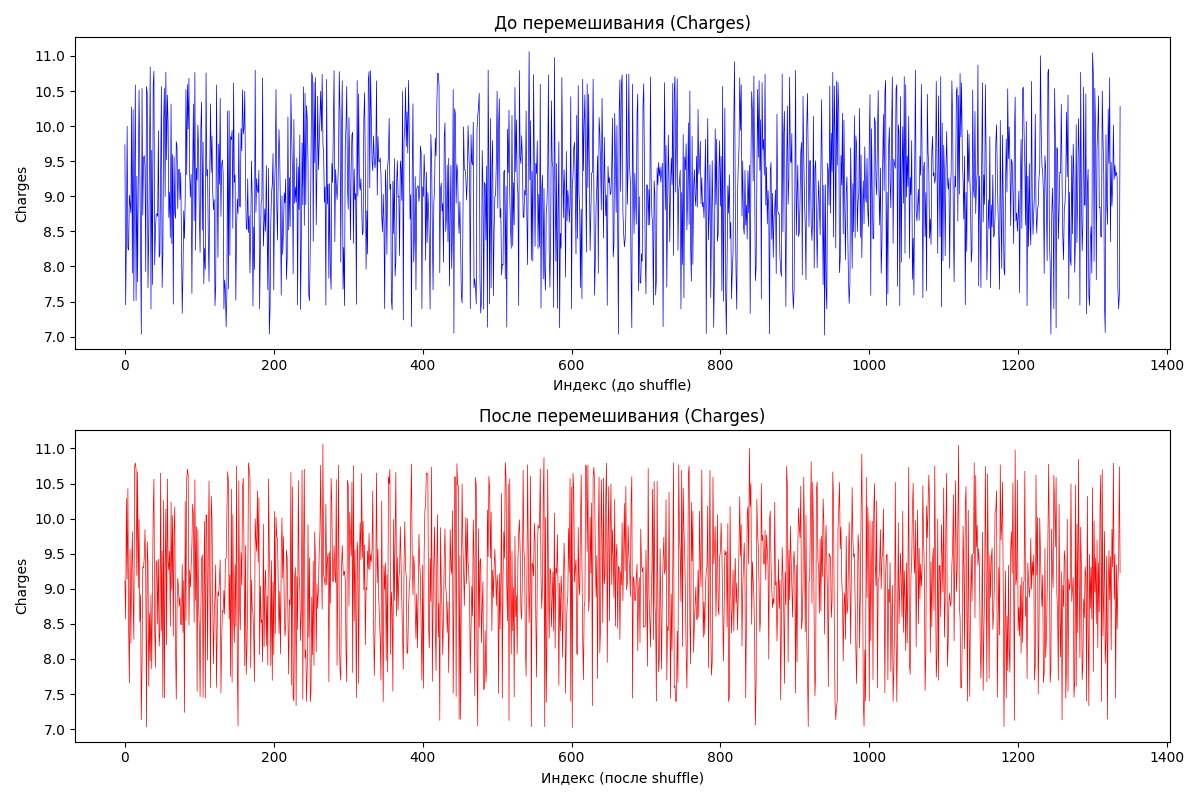
* **Логарифмическое преобразование целевой переменной.**  
  Преобразование выполняется командой:

df['charges'] = np.log(df['charges'])

Такое преобразование уменьшает влияние выбросов и приводит распределение расходов к более нормальному виду.

* **Перемешивание и разбиение данных.**  
  Данные перемешиваются и разбиваются на обучающую (70%) и тестовую (30%) выборки:
* **Визуализация этапа предобработки.**  
  Для контроля корректности перемешивания и нормализации генерируются графики:
  + **Эффект перемешивания:**  
    Линейные графики до и после перемешивания сохраняются в файле shuffle\_effect\_lineplot.png.
  + **Распределение признаков до и после нормализации:**  
    Гистограммы для признаков, например, age и bmi, сохранены в файле preprocessing\_effect.png.



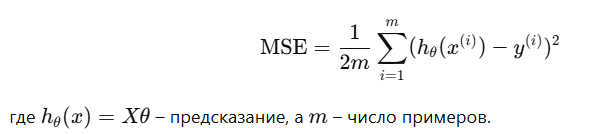


# **2. Реализация регрессии двумя способами**

## 2.1 Модель без использования библиотек

В файле model\_no\_libs.py реализована линейная регрессия «с нуля» с использованием метода градиентного спуска. Основные этапы:

* **Добавление столбца смещения (bias).**  
  Для учета свободного члена к матрице признаков добавляется столбец единиц.
* **Инициализация параметров.**  
  Вектор параметров theta инициализируется нулями.
* **Определение функции ошибки (MSE) и вычисление градиента.**  
  Функция ошибки определяется как:



Код функции:

def mse(X, y, theta):m = len(X)  
 predictions = mat\_mult(X, theta)  
 error = matrix\_subtract(predictions, y)  
 squared\_error = [[val[0] \*\* 2] for val in error]  
 return sum(row[0] for row in squared\_error) / (2 \* m)

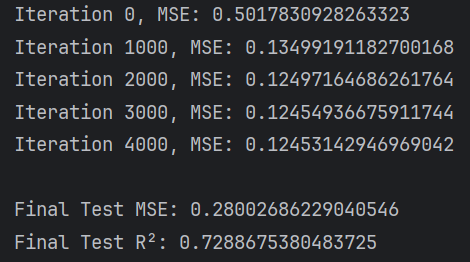
Аналогично определяется функция для вычисления градиента:

def gradient(X, y, theta, lambda\_reg=0.01):  
 m = len(X)  
 predictions = mat\_mult(X, theta)  
 error = matrix\_subtract(predictions, y)  
 X\_T = transpose(X)  
 grad = mat\_mult(X\_T, error)

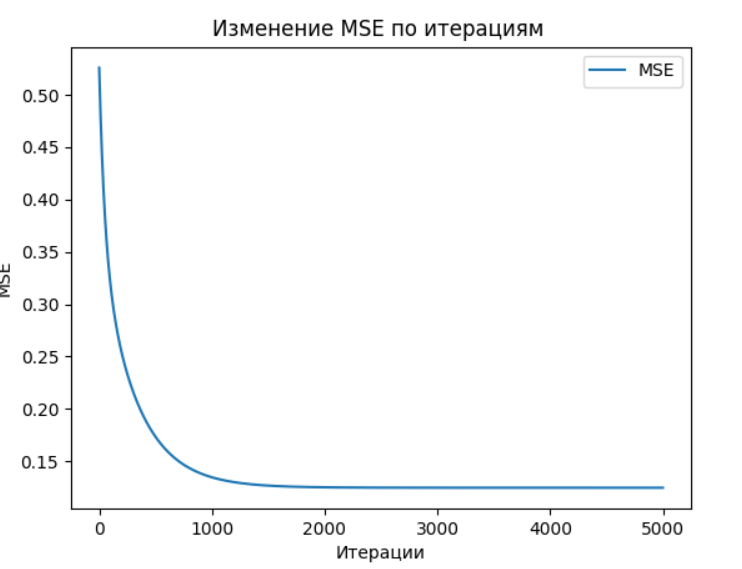
* **Градиентный спуск.**  
  В цикле происходит обновление параметров:

for i in range(num\_iterations):  
 grad = gradient(X\_train\_bias, y\_train, theta)  
  
 for j in range(len(theta)):  
 theta[j][0] -= learning\_rate \* grad[j][0]  
  
 cost = mse(X\_train\_bias, y\_train, theta)  
 mse\_history.append(cost)  
  
 if i % 1000 == 0:  
 print(f"Iteration {i}, MSE: {cost}")

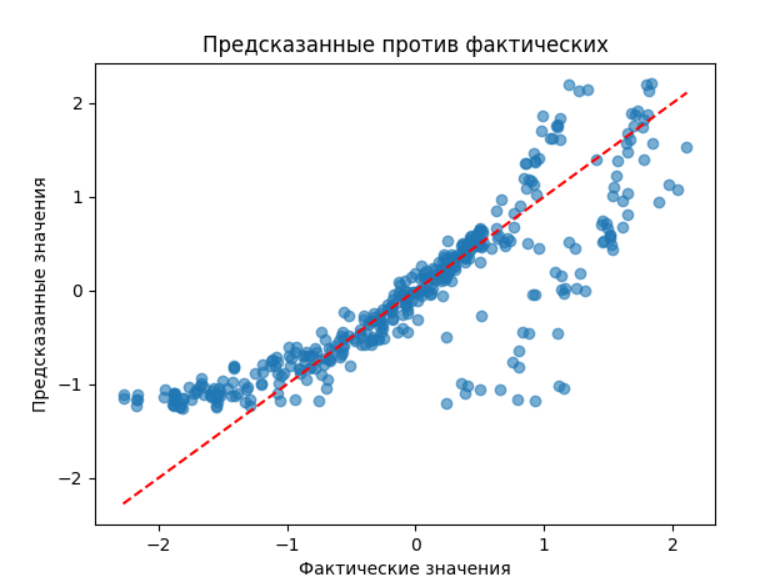
Каждые 1000 итераций выводится значение MSE для контроля процесса обучения.



* **Оценка качества модели.**  
  После обучения вычисляются показатели MSE и коэффициент детерминации R^2 для тестовой выборки. Для визуализации результатов генерируются:
  + График изменения MSE по итерациям (no\_libs\_mse\_vs\_iteration.png).



* + График «Предсказанные против фактических» значений (predicted\_vs\_actual\_no\_libs.png).



## 2.2 Модель с использованием библиотек

Файл model\_libs.py демонстрирует применение библиотеки scikit-learn для построения модели линейной регрессии:

* **Загрузка данных.**  
  Данные считываются из файлов train\_numeric.csv и test\_numeric.csv, где уже произведена предварительная обработка.
* **Обучение модели с помощью LinearRegression.**

model = LinearRegression()

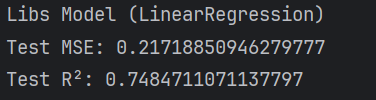
model.fit(X\_train, y\_train)

Этот готовый инструмент автоматически учитывает наличие смещения.

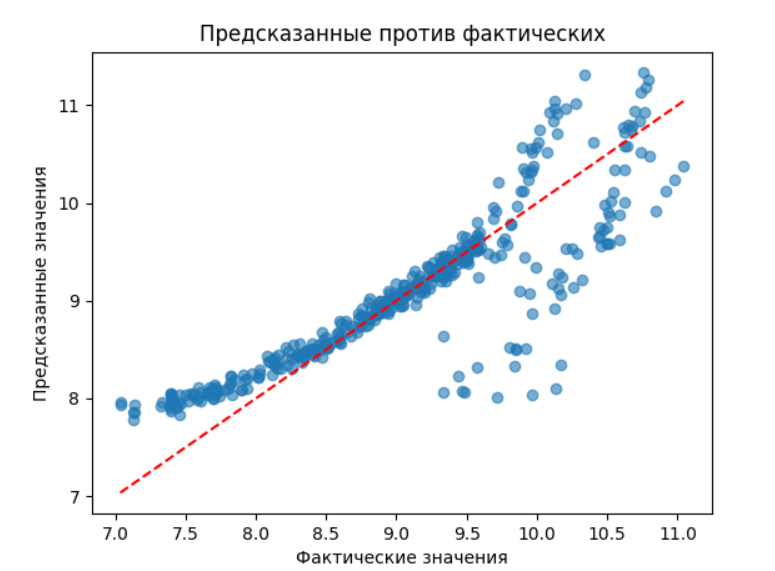
* **Оценка качества модели.**  
  Предсказания на тестовой выборке сравниваются с истинными значениями с помощью метрик MSE и R^2:

mse\_test = mean\_squared\_error(y\_test, y\_pred\_test)

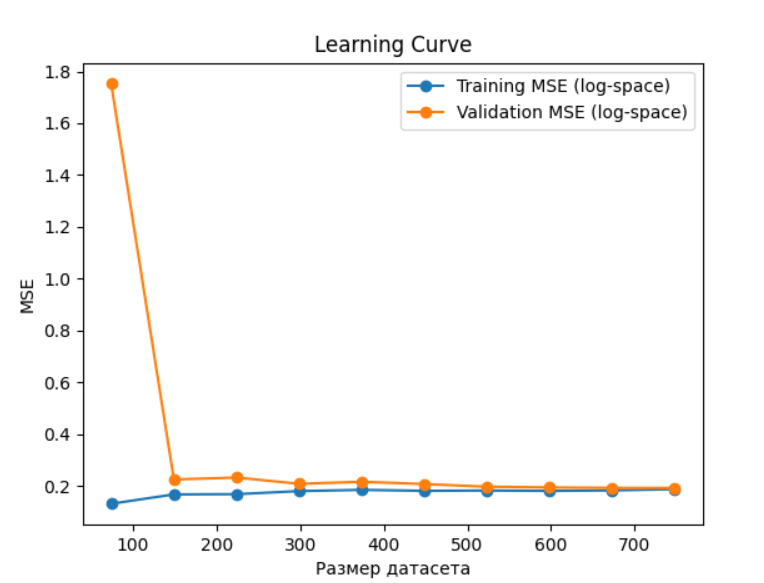
r2\_test = r2\_score(y\_test, y\_pred\_test)



* **Визуализация результатов.**  
  Для данной модели генерируются следующие графики:
  + **Predicted vs. Actual:**  
    Точечный график, показывающий соответствие предсказанных и фактических значений (файл predicted\_vs\_actual\_libs.png).



* + **Learning Curve:**  
    График зависимости MSE от размера обучающей выборки.



# **3. Разработка программы для работы с обученной моделью**

Файл predict\_ui.py реализует графический интерфейс для взаимодействия с пользователем. Основные моменты реализации:

* **Предварительные расчёты на базе исходных данных.**  
  Считываются исходные данные из файла insurance.csv для вычисления статистических параметров (медианы, средних значений и стандартных отклонений), которые используются для нормализации входных данных.
* **Кодирование категориальных признаков.**  
  Значения для признаков sex, smoker и region преобразуются в числовые с использованием заранее определённых отображений.
* **Загрузка параметров обученной модели.**  
  Из файла theta\_libs.txt загружаются коэффициенты, полученные в модели с использованием библиотек.

theta\_libs = np.loadtxt("theta\_libs.txt")

* **Функция предсказания.**  
  Функция predict\_charge принимает сырые входные данные, проводит их нормализацию и кодирование, формирует вектор признаков (с учетом смещения) и вычисляет логарифмическое предсказание, которое затем преобразуется в значение в долларах:

def predict\_charge(raw\_age, raw\_sex, raw\_bmi, raw\_children, raw\_smoker, raw\_region):

norm\_age = (raw\_age - mean\_age) / std\_age

norm\_bmi = (raw\_bmi - mean\_bmi) / std\_bmi

norm\_children = (raw\_children - mean\_children) / std\_children

mapped\_sex = sex\_map[raw\_sex.lower()]

mapped\_smoker = smoker\_map[raw\_smoker.lower()]

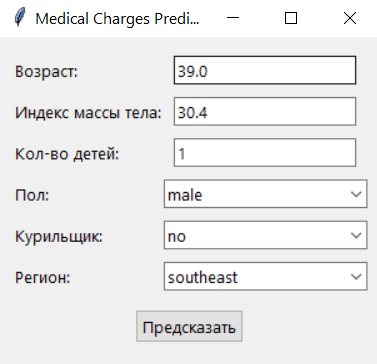
mapped\_region = region\_mapping[raw\_region]

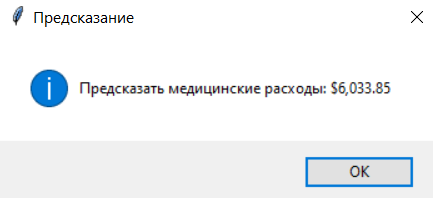
x = np.array([1.0, norm\_age, mapped\_sex, norm\_bmi, norm\_children, mapped\_smoker, mapped\_region], dtype=float)

log\_pred = np.dot(theta\_libs, x)

return np.exp(log\_pred)

* **Интерфейс пользователя.**  
  С использованием библиотеки tkinter создается окно с полями для ввода значений (возраст, BMI, количество детей, выбор пола, курильщика и региона). При нажатии кнопки «Предсказать» данные обрабатываются, и пользователю выводится предсказанное значение расходов.





# **ЗАКЛЮЧЕНИЕ**

В отчете описан комплексный подход к задаче предсказания страховых расходов. Были реализованы две модели:

* **Без использования библиотек:** модель, реализованная вручную, демонстрирует принцип работы градиентного спуска и позволяет глубже понять внутренние механизмы линейной регрессии.
* **С использованием библиотек:** применение scikit-learn позволяет быстро построить модель и получить готовые решения для оценки качества, а также визуализировать результаты.

Кроме того, разработанный графический интерфейс делает систему доступной для конечного пользователя, позволяя вводить данные и получать предсказания в режиме реального времени.

Предоставленные визуализации (например, графики предсказанных против фактических значений, learning curve и эффекты предварительной обработки) помогают оценить качество моделей и увидеть влияние этапов обработки данных на итоговый результат. Такой комплексный подход даёт возможность не только построить модель, но и понять все нюансы её работы, а также выявить возможные направления для дальнейшего улучшения.